Terascale Turbulence Computation on BG/L Using the FLASH3 Code

R.T. Fisher[1,2,8], L. Kadanoff[3,4], D.Q. Lamb[1,2], P. Constantin[4], I. Foster[5,6], F. Cattaneo[1,2], M.E. Papka[1,5,6], A. Dubey[1,2], T. Plewa[1,2], P. Rich[1], K. Antypas[1], S.I. Abarzhi[1], S.M. Asida[1,7], A.C. Calder[1,2], L.B. Reid[1], D. Sheeler[1]. J.B. Gallagher[1], C.C. Glendenin[1], S.G. Needham[1]

ABSTRACT

Understanding the nature of turbulent flows remains one of the outstanding questions in classical physics. Significant progress has been made recently using computer simulation as an aid to our understanding of the rich physics of turbulence. Here we present both the computer science and scientific features of a unique terascale simulation of a weakly-compressible turbulent flow, including tracer particles. The simulation was performed on the world's fastest supercomputer as of March 2007, the Lawrence Livermore National Laboratory IBM BG/L, using version 3 of the FLASH code. FLASH3 is a modular, publically-available code, designed primarily for astrophysical simulations, which scales well to massively parallel environments.

We discuss issues related to the analysis and visualization of such a massive simulation, and present initial scientific results. We also discuss the opening of the dataset and challenges related to its public release. We suggest that widespread adoption of an open dataset model of computing is likely to result in significant gains for the scientific computing community in the near future, in much the same way that the widespread adoption of open source software has produced similar gains in the last decade.

---

[1]Center for Astrophysical Thermonuclear Flashes, The University of Chicago, Chicago, IL 60637

[2]Department of Astronomy and Astrophysics, The University of Chicago, Chicago, IL 60637

[3]Department of Physics, The University of Chicago, Chicago, IL 60637

[4]Department of Mathematics, The University of Chicago, Chicago, IL 60637

[5]Department of Computer Science, The University of Chicago, Chicago, IL 60637

[6]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439

[7]Racah Institute of Physics, Hebrew University, Jerusalem, Israel

[8]Contact author: rtfisher@uchicago.edu (773) 834-3904

1. Introduction

Turbulent flows are ubiqutious in nature, arising in scales as small as table-top experiments of fluids to scales as large as interstellar gas clouds, and play a fundamental role in the mixing, transport, and combustion of fluids. Yet despite its importance, turbulence largely remains an unsolved problem, particularly when one is interested in computing detailed properties of a flow, such as the turbulent drag over an airplane wing, or the rate of combustion in a turbulent flame.

The last two decades have seen our knowledge of turbulent flows grow in leaps and bounds, thanks in no small part to the use of computers both in the laboratory and in simulation. Beginning in December 2005, the ASC Alliances Center for Astrophysical Thermonuclear Flashes at the University of Chicago was one of six groups in all fields of computational science invited to use the Lawrence Livermore National Laboratory (LLNL) BG/L supercomputer (currently the world's largest and fastest), shortly before it was permanently incorporated into their secure network. The Flash Center's simulation is the world's largest weakly-compressible, homogeneous, isotropic simulation in existence (5; 15; 16). Approximately one week of CPU time on 65,536 processors in coprocessor mode was used to complete the simulation. The turbulence was also tracked with 256^3 dimensionless Lagrangian particles. Our findings indicate that the wealth of data gathered is of very high quality, and can be used to constrain fundamental theories of turbulence, as well as provide a "virtual turbulence laboratory" in which many other ideas (such as subgrid models of turbulence) can be tested. We will open access to this dataset publicly later this year.

The FLASH code (7; 13) used in this turbulence simulation is a modular, component-based application code used for simulating compressible, reactive flows found in astrophysical environments. The code supports multiple methods for managing the discretized simulation mesh, including an in-house Uniform Grid implementation, and the PARAMESH library (19) which implements a block-structured adaptive grid. The FLASH code scales very well and was chosen in the spring of 2004 as one of the marquee applications for the new BG/L platform (6). The BG/L machine revives the massively parallel computing paradigm of large numbers of relatively slow and simple processors with limited memory (10). It provides teraflop-scale computation through duplication of hardware, rather than through hardware complexity. In its largest incarnation at LLNL, the parallelism is an order of magnitude larger than any other available platform. BGL's configuration is a marked deviation from the prevailing practices in high performance computing, and presents both opportunities and challenges to application developers and users. Because each BG/L processor offers lower performance and less memory than processors on other platforms, scientific applications attempting to effectively use this platform must be able to exploit fine-grained parallelism while scaling to many tens of thousands of processors.

2. Architecture

The most recent major release of the FLASH code is version 3. The FLASH3 architecture is defined by two entities: the units, and the setup tool (3). FLASH3 is neither a single application nor a single binary. Rather, it is a collection of units combined by the setup tool into an application tailored for specific simulations. The unit is the basic functional entity in FLASH3; it provides well-defined functionality and conforms to a structure that facilitates interactions with other units. All units must conform to a set of inheritance and encapsulation rules. For example, a unit may have sub-units to collect self-contained subsets of its functionality into easily identifiable entities. A unit may also have interchangeable implementations of varying complexity, including child implementations that inherit from and override the functionality of the parent.

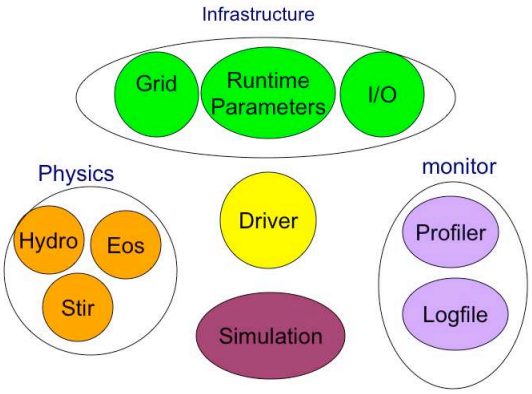

Fig. 1.— Examples of FLASH3 units.

FLASH units can be broadly classified into five groups: *infrastructure, physics, monitor, driver,* and *simulation* (see Figure 1). The *infrastructure units* are responsible for all the housekeeping tasks of the code such as managing the grid which describes the physical domain, making runtime parameters available as needed, and organizing all the input and output (I/O) from the code. The *physics* units implement algorithms to solve the mathematical equations describing specific physical phenomena, for example *Hydro* (for hydrodynamics) and *EOS* (for the equation of state) (8). The *monitoring* units *Logfile* and *Profiler* track the progress of an application, while the *Driver* unit implements the time advancement methods and controls the simulation by managing the interaction between the included units. The *Simulation* unit is of particular significance because it specifies a given application by defining the initial conditions and simulation-specific runtime parameters.

Each unit has one or more text files called *Config* files. These specify the (sub)unit requirements, such as physical variables, applicable runtime parameters, and the interaction with other code units. The setup script starts building an application by first examining the Config file in the Simulation unit, which specifies units essential for the simulation. The setup

tool then parses the Config files of required units to find their physical variables, runtime parameters and necessary libraries to create a cohesive set of files defining an application. Finally, the user compiles the code pieces to produce an executable application for his problem. A user can replace any native FLASH routine by providing an alternative implementation in the problem setup directory. For example, users can provide their own criteria for refining the mesh by including the appropriate code files with compatible function interfaces in the Simulation directory.

3. Algorithms

The large-scale turbulence simulation described in this paper used a uniform Cartesian grid, which divides the physical domain into uniformly-spaced computational cells. Periodic boundary conditions are applied to all physical variables at the domain edges. Individual computational cells are grouped into equal-sized cubical subdomains, which are then mapped to different processors. We refer to these subdomains as blocks. A perimeter of non-physical guardcells surrounds each block of data, providing it either with data from the neighboring blocks (if the block is in the computational domain interior proper), or from boundary conditions (if the block lies on the edge of the computational domain).

The hydrodynamics solver used in the simulation is a directionally-split piecewise-parabolic method (PPM) (9) that solves the Eulerian equations with an explicit second-order accurate forward time difference (13). Time-advanced fluxes at cell boundaries are computed using the numerical solution to the Riemann problem at each boundary. Initial conditions for each Riemann problem are determined by assuming the nonadvanced solution to be piecewise-constant in each cell. Using the Riemann solution has the effect of introducing explicit nonlinearity into the difference equations of flow and permits the calculation of sharp shock fronts and contact discontinuities without introducing significant nonphysical oscillations into the hydrodynamics. The flow variables are represented with piecewise parabolic functions, making this scheme a second order accurate method.

Because theories of turbulence generally assume a steady state, and because turbulence is inherently a dissipative phenomenon, we have chosen to drive the simulation to sustain a steady-state. This driving must be done carefully in order to avoid introducing artifacts into the turbulent flow. We use a relatively sophisticated stochastic driving method originally introduced by Eswaran & Pope (11). The turbulent velocity fluctuations are described by Fourier-transforming from the spatial domain. For each "stirred" mode of the velocity field, an acceleration is applied at each timestep. The field consists of three complex phases, with each acceleration mode evolved by an Ornstein-Uhlenbeck (OU) random process, a process which is analogous to Brownian motion in a viscous medium. An OU process is a time-correlated, zero-mean, constant-root-mean-square process. Each next step in the process

adds a Gaussian random variable with a given variance, weighted by a "driving" factor $\sqrt{(1-f^2)}$, where $f = e^{-\frac{\Delta t}{\tau_{\rm decay}}}$ then "decays" the previous value by an exponential factor f. Since the OU process represents a velocity, the variance is chosen to be the square root of the specific energy input rate divided by the decay time $\tau_{\rm decay}$. In the limit that the timestep $\Delta t \to 0$, the algorithm represents a forcing term that is a linearly-weighted summation of the old state with the new Gaussian random variable.

By evolving the phases of the stirring modes in Fourier space, imposing a divergence-free condition is relatively straightforward. At each timestep, the solenoidal components of the velocities are projected out, leaving only the non-compressional modes to add to the velocities. The velocities are then converted to physical space by a direct Fourier transform – adding the trigonometric terms explicitly. Since the stirring is done in the low modes, most drivings involve a fairly small number of modes; therefore this decomposition is more efficient than a complete Fast Fourier Transform. The FFT would need large numbers of modes (six times the number of cells in the domain), the vast majority of which would have zero amplitude.

The simulation also evolved the movement of massless tracer particles, which are point-like objects characterized by positions $\mathbf{x}_i$, and velocities $\mathbf{v}_i$. The characteristic quantities of each particle are defined by the position of the particle and are determined by interpolation from the grid mesh. The particles move with the flow relative to the mesh and can travel from block to block, requiring communication patterns different from those used to transfer boundary information between processors for mesh-based data. The implementation in FLASH directionally splits the movement of particles out of a block. Consider a particle that moves from (x, y, z) to (x_1, y_1, z_1). If the new coordinate (x_1) is outside the particle's current block, it is moved to the appropriate neighbor along the first dimension. The particle is now owned by the neighbor, and when examining movement along the second dimension, the neighboring block will treat it identically to its own particles. The processes is repeated for all the dimensions, until the particle terminates movement in the correct block. The direction splitting halves the number of explicit data exchanges between processors per timestep from 26 (potentially one exchange with every neighbor including those along the corners) to 13. The computation required for a particle is performed by the same processor that evolves the block in which the particle resides. No effort is made to separately load-balance the particle computations because experience has shown that the cost of load balancing the particles outweighs the gain of maintaining a good load balance for what is a small fraction of the overall execution time. The effect of this choice on scaling is discussed below. The time integration used for the Lagrangian particles in this simulation uses a predictor-corrector scheme that yields a solution better than first-order accurate if the timesteps frequently change, and a second-order accurate solution if they stay relatively uniform (3).

4. Challenges of Scale

The BG/L machine deviates from the current parallel computing norm of relatively few but very powerful processors with large memory. This standard model derives from cluster-based computing, which is driven by single processing element (PE) performance. The BG/L machine, on the other hand, uses slower processors with limited memory, and relies upon finer grain parallelism to achieve performance. This shift in the computing paradigm presents major challenges to codes that are memory- intensive and have components with global communication patterns.

Multiphysics codes like FLASH3 usually face trade-offs between modularity, efficiency of memory, and CPU usage. For example, in order to keep various physics solvers in the code independent of one another, and their data management well modularized, multiple copies of some datasets may exist in the code. This problem could be alleviated with dynamic memory management, but that solution causes loss of compiler optimizations and therefore computational efficiency. Also, while a single timestep evaluation typically involves nearest neighbor communications only, some operations are necessarily global, such as updating the timestep selected by the Courant condition. Additionally, scientific calculations generate massive amounts of data, and the I/O management and post-processing usually becomes the biggest challenge.

Because turbulent flow simulations benefit from high resolution everywhere, the simulation described here did not need to use the adaptive mesh capabilities of the FLASH code (but see also (18)). Use of a uniform grid reduced the global communication complexity of the Eulerian part of the solution, letting it scale almost perfectly, as shown in the weak scaling plot of Figure 2. Figure 2 plots time taken to advance the solution by 50 timesteps; both axes are in logscale. The two lines show the overall evolution time and the time taken by the tracer particles. The number of processors along the x-axis grows from 1 to $32,768$; and the amount of work grows in proportion to the number of processors. The Lagrangian tracer particles, however, do not show the same perfect scaling, even though their communications pattern is nearest-neighbor like that of the Eulerian grid. As mentioned above, no separate effort is made to load-balance the particles. The complex nature of the turbulent flow changes the particle distribution during the evolution, which unbalances the particle load distribution and degrades the scaling. But because the particles account for only a small fraction (less that 0.3%) of the overall execution time, the effect on the overall scaling is negligible.

Despite the excellent overall scaling, the sheer scale of the simulation and the tremendous amount of generated output data still presented a huge challenge. We took particular care before the run to test the scaling of FLASH3 I/O, given the lack of evidence of successful parallel I/O on the BG/L machine or on other such large scale computations. As anticipated, we found that none of the parallel I/O libraries available with FLASH3, namely HDF5 (20),

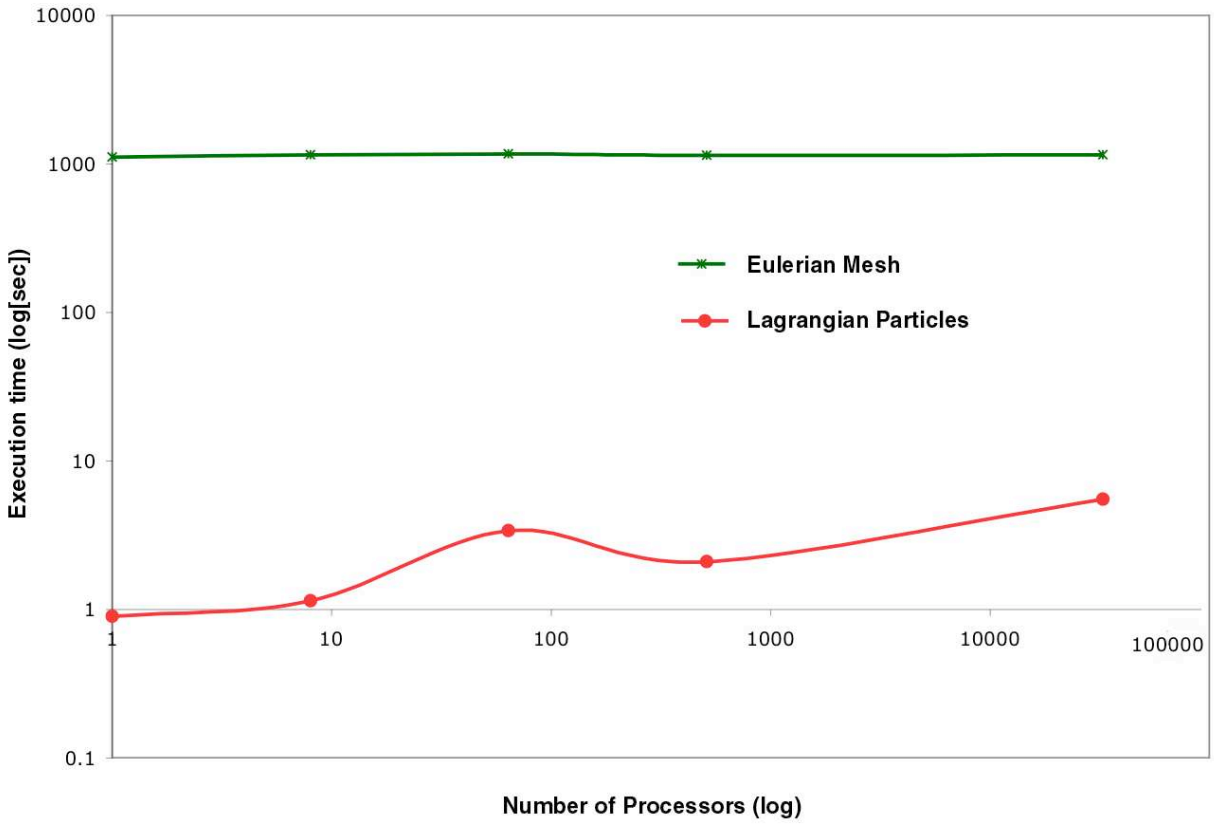

Fig. 2.— Weak scaling of the Eulerian mesh and Lagrangian particles with FLASH3 for the driven turbulence problem.

PnetCDF (1), or basic MPI-IO (2) scaled to more than 1024 processors. This limitation required us to implement a direct I/O approach whereby each process wrote its portion of the global data to its own file in the Fortran sequential format. In all, the run totaled about 2,300 separate data dumps in time, and each dump created 32,768 files, for a total of roughly 74 million files, occupying 154 TB of disk capacity. Here, the challenge was not just the volume of data, but also the almost unmanageable number of files. The transfer of data to the local site for analysis took several weeks, using four nodes of the LLNL machine ALC, each running gridftp (24). The total sustained transfer rate averaged about 20 MB/sec. We are now using computing clusters to visualize and analyze the data in small sections at a time, which has become a large-scale computing project of its own.

5. Science

5.1. Introduction

The single most significant contribution to turbulence theory was Kolmogorov's 1941 idea that at very high Reynolds numbers, turbulent flows develop an energy cascade from large scales down to small which is governed by self-similarity (17). The range of scales between the large modes where the fluid is driven, and the small scales on which its energy is dissipated, is known as the inertial regime. As a consequence of self-similarity at high Reynolds numbers. Thus, according to Kolmogorov, the high-Reynolds number turbulent flow is said to be "universal": the resulting state of any two such fluids are governed by the same set of scaling laws in the inertial regime.

Over the past twenty years, extensive research has been performed on turbulent flows. One of the key insights attained in both experiment and in numerical simulation is that Kolmogorov's original theory is incomplete: the hypothesized self-similarity is broken because the dissipation within the fluid does not occur homogeneously, but instead is intermittent in both space and time. The resulting scaling laws, which differ from Kolmogorov, are said to exhibit "anomalous scaling."

Kolmogorov's theory is to turbulence what Newton's laws are to mechanics: a highly successful, though incomplete framework whose enormous influence is widely felt. A number of phenomenological theories have been advanced as possible heirs to Kolmogorov's legacy, though it remains unclear which of these (if any) is correct (12; 23). Working directly from the Navier-Stokes equations presents challenging barriers to standard mathematical analysis techniques, so theorists must instead work from plausible assumptions.

Another issue is that our techniques of treating hydrodynamical flows away from the strictly incompressible limit is different than the majority of work done in the turbulence community. Hence, we also need to compare our results against experiment, theory, and previous simulations in order to convince ourselves that weak compressibility does not have a strong influence upon the scientific conclusions we draw from our dataset.

In the following, we use the idea of anomalous scaling to probe our BG/L turbulence run dataset to gather clues to these questions.

5.2. Results

A fundamental mathematical tool commonly used by turbulence researchers to examine the scaling properties of turbulent flow is the pth-order structure function $S_p(r)$, which is in fact closely related to the autocorrelation function of the velocity field :

$$S_p(r) = \langle |\vec{v}(\vec{x}+\vec{r}) - \vec{v}(\vec{x})|^p \rangle \propto r^{\xi_p}$$

Here $\langle\rangle$ denotes an average of spatial locations $\vec{x}$ over all space. The proportionality on the right-hand side applies over a range of separations r in the inertial regime, and is a direct consequence of self-similarity.

A naive computation of $S_p(r)$ involves an average over all spatial points. However, instead of calculating $S_r(p)$ directly, we generate probability distribution functions (PDFs) of each component of the velocity increment $\Delta v_r = \vec{v}(\vec{x}+\vec{r}) - \vec{v}(\vec{x})$ of each interval r for each dimension for each dimensional velocity component. Since this approach only calculates the separations along a given dimension, it allows for a very natural slab-decomposition of the domain of a given timestep, as well as very straightforward parallelization, enabling us to perform this analysis on a small cluster. Once the PDFs of velocity increments are generated, the structure functions can be computed directly, from

$$S_p(r) = \int d\Delta v_r P(\Delta v_r) \Delta v_r^p$$

The Lagrangian tracer particle data can be analyzed in a similar fashion, taking the pth order structure function with respect to time:

$$S_p(\tau) = \langle |\vec{v}(t+\tau) - \vec{v}(t)|^p \rangle \propto \tau^{\xi_p}$$

While the Lagrangian dataset is significantly smaller, problems in the calculation do arise as each timestep is contained in a different file, making I/O much more costly due to the frequency of accesses than it is with the Eulerian dataset. A similar method of histogram generation is utilized as is described above. The structure-function data generated from this set was then used to verify the Lagrangian tracer particle data.

A very useful technique in determining the scaling exponents ξ_p was first discovered by Roberto Benzi and colleagues, who noted that an extended self-similar region appears when, instead of plotting the pth-order structure function versus r, one plots versus the third-order structure function (4). This technique is referred to as *extended self-similarity* (ESS). The scaling exponents derived from ESS are in fact identical to the ξ_p above, since the third-order structure function $S_p(r)$ is exactly proportional to r, as was first rigorously demonstrated by Kolmogorov in his famed 4/5th law. The advantage of ESS is that the self-similar scaling regime is very broadly extended over 1-2 more decades in lengthscale, which permits much more precise determinations of the scaling law exponents.

One can easily demonstrate that, were Kolmogorov's original theory strictly correct, and if dissipation occurred homogeneously, the exponents ξ_p would simply equal $p/3$. In

the inertial regime, the flow is self-similar and no other characteristic length scales are introduced. Therefore, the scaling exponents ξ_p follow directly from simple dimensional analysis: assuming a characteristic velocity v over a spherical region of size r, the dissipation per unit mass ϵ within r must scale as $\epsilon \propto v^3/r$. If ϵ is homogeneous, and the flow is steady, one infers that $v \propto r^{1/3}$. Dimensional analysis then reveals the exponents ξ_p are simply equal to $p/3$.

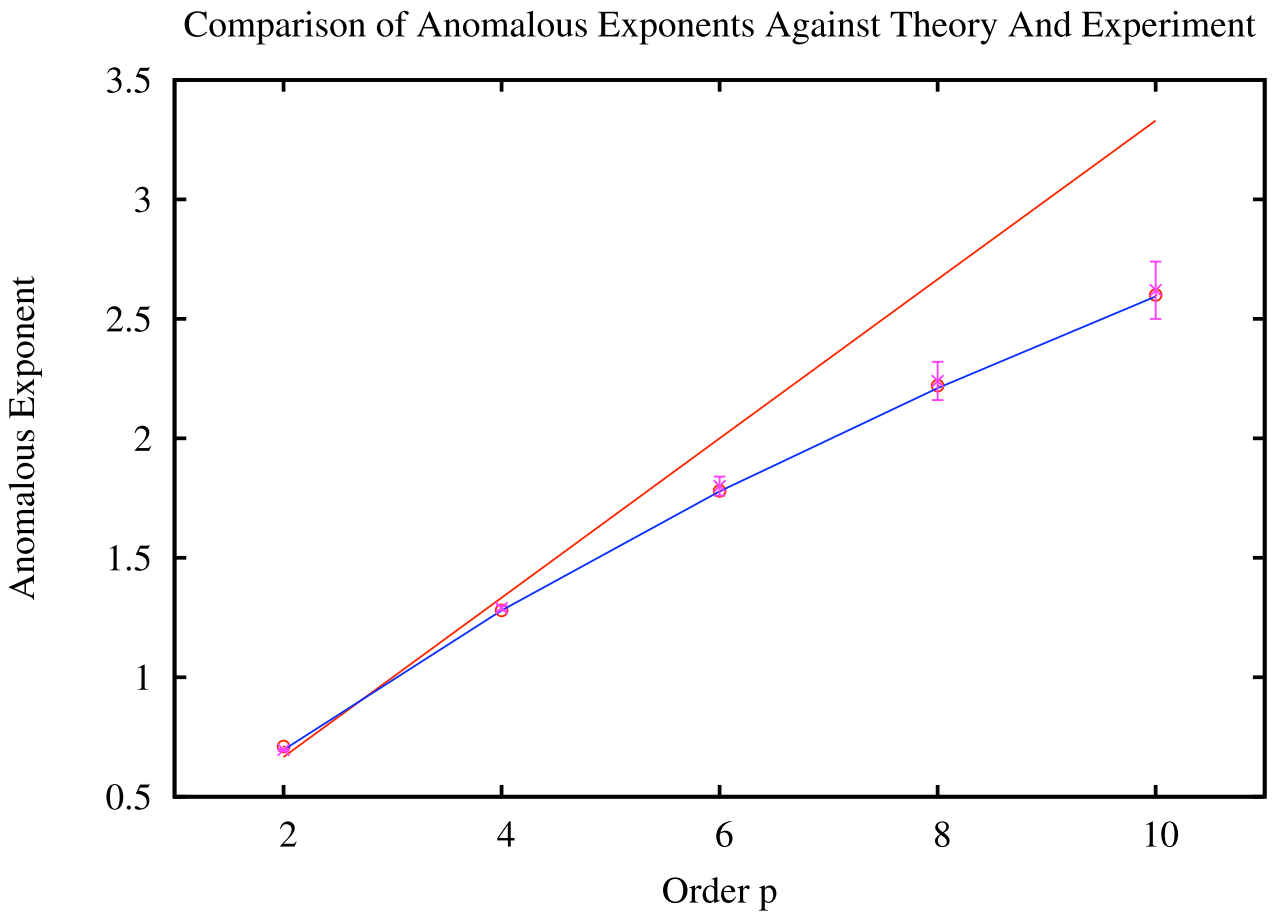

Fig. 3.— Anomalous scaling exponents (through $p = 10$) from the BG/L run (points with error bars) compared against predictions of Kolmogorov's original 1941 theory (straight solid line), as well as the predictions of She-Leveque (curved solid line), and experimental results analyzed by Benzi (solid circles).

We can directly test Kolmogorov's original theory, as well as more recent theories, by applying ESS to the data from the BG/L run. The results are shown in in Figure 3. These results show the scaling exponents ζ_p versus p derived from fits of the structure function as a function of separation r along x – one of the three principal components of the *longitudinal structure function*. As is easily seen, Kolmogorov's 1941 theoretical prediction deviates systematically from both Benzi's experimental analysis and our simulation. In contrast,

the more recent theory of She-Leveque's (23) is in close agreement with both experiment and simulation. We conclude that despite being weakly-compressible, our simulation agrees very well with previous incompressible results, and suggests that our dataset can be used to explore a wide variety of issues in turbulent flows.

5.3. Visualization

Current visualization efforts have focused on exploration of the dataset. This effort has utilized a combination of open source solutions (e.g., ParaView, VisIt) and specific solutions to meet analysis needs. Processing has been done to augment the dataset with post-simulation content such as scalar magnitudes of the local vorticity and divergence vectors, with this data now being stored as part of the dataset that will be made publicly-available. Visualization challenges have demonstrated the need for large resources beyond the initial simulation, the need for data filtering and selection tools to reduce the data to appropriate amounts for viewing, and investment in remote visualization solutions. The generation of the derived data fields for vorticity and divergence required 1.5 days on a 64 node cluster. Visualizing all 16 million Lagrangian particle traces yields no useful insight, instead filtering and cutting the data to highlight features and areas of interest is crucial (see Figures 4 and 5). Future efforts will involve the integration of work-flow tools for the construction of analysis pipelines, the continued augmenting of the data to increase its value to the community, and the availability of focused tools for users to visualize the data.

6. Open Dataset Model

The public release of the dataset will perhaps be one of the most far-reaching consequences of this effort. Performing simulations at the scale of the FLASH run is by necessity a major challenge that few research efforts have both the technical capacity and computational resources to address. Even when successful, most such datasets are not typically publicly-released, which limits their impact to a tiny set of researchers privy to the data. Here we draw inspiration from the open source movement, and advocate an open model for high-performance computing datasets – namely, that large-scale high-performance computing datasets should be made openly-accessible. While dissemination and analysis of high-performance computing datasets poses numerous technical challenges, we argue that these technical challenges can be addressed and overcome, and are outweighed by the potential scientific gains to be achieved by sharing the datasets.

Historically, the movement known today as "open source" originated with the IBM user group SHARE in the 1950s, and was later championed by Richard Stallman and the Free Software Foundation beginning in the early 1980s. However, it first received widespread at-

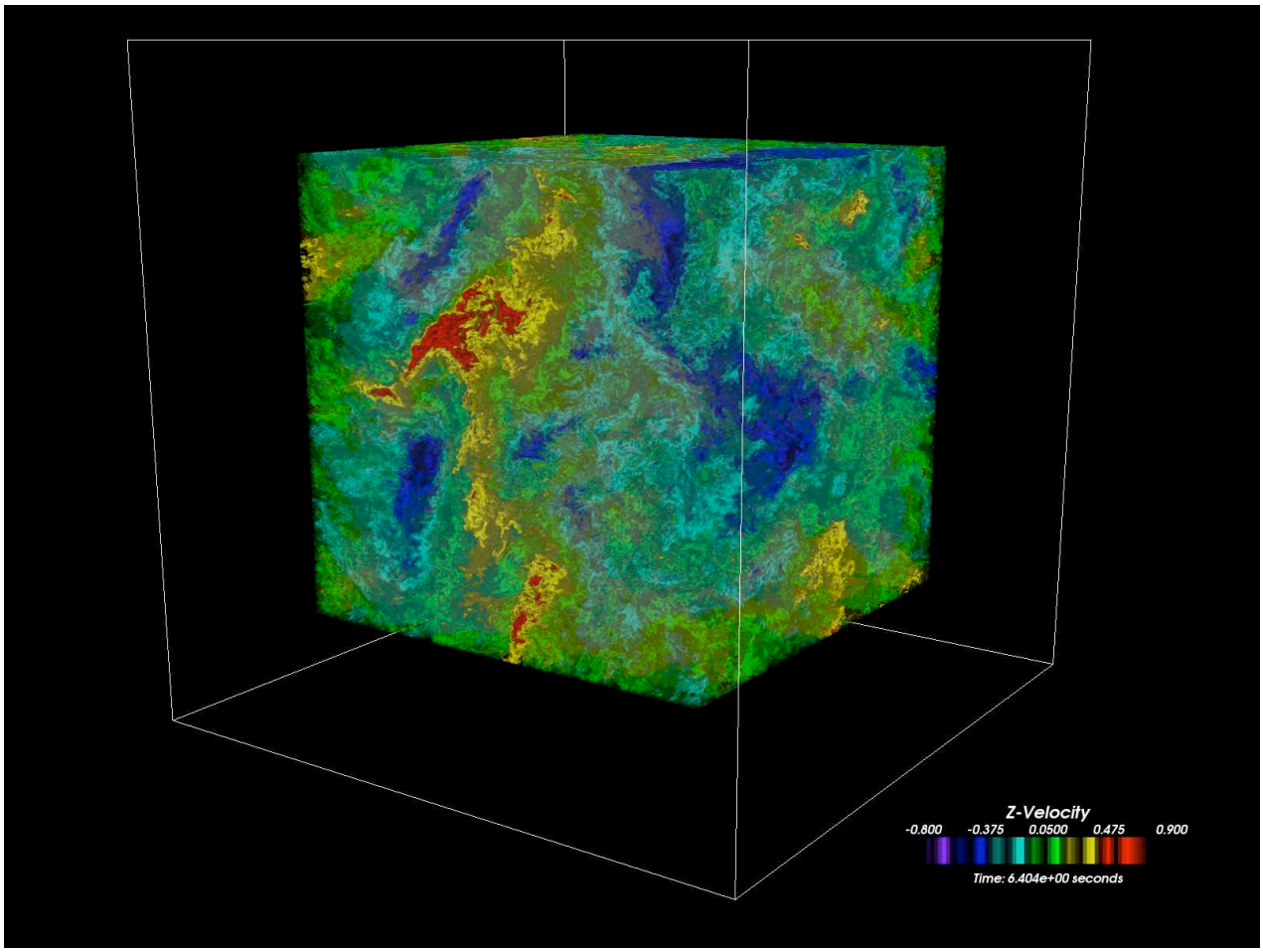

Fig. 4.— Volume rendering of one component of velocity of the grid data in fully-developed turbulence, taken over a subset of the domain.

tention following Eric Raymond's highly-influential paper, "The Cathedral and the Bazaar," (22) which inspired Netscape to release a version of its browser software in 1997.

The open source model of distributing source code has had an enormous impact on computing in both the scientific and business worlds. At the heart of the open source paradigm is the elimination of barriers to the exchange of source code, thereby creating a free marketplace of ideas wherein communication and resource sharing is encouraged. In the years following Raymond's essay, the number of open source code development projects skyrocketed, and companies such as IBM, HP, Oracle, and CollabNet have incorporated the open source paradigm into their business models. Perhaps even more far-reaching are the wide array of other endeavors inspired by the open source paradigm – including everything from open publication projects such as Wikipedia, The Creative Commons, and The Science Commons to open educational curricula such as the OpenCourseWare project at MIT, and numerous projects in between.

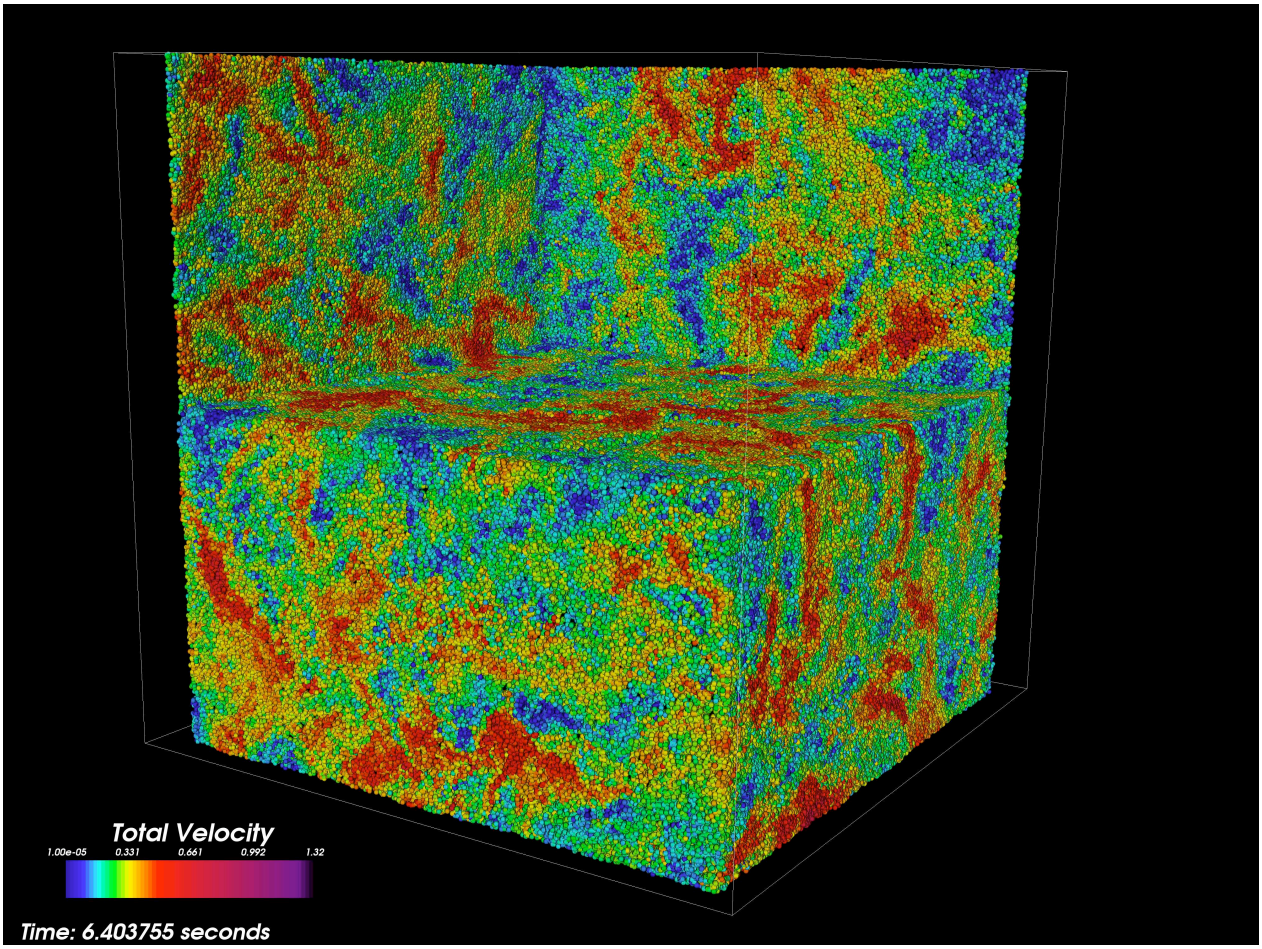

Fig. 5.— Particles contained within cut planes of the problem domain in fully-developed turbulence, shaded by velocity.

We believe that the same open-minded thinking which gave rise to the open source model of distributing source code can also be profitably applied to large-scale high-performance computing datasets. In other areas of science where experimental and observational datasets have been made open to researchers, enormous strides have been made. Particularly-successful example include the Human Genome Database (14), the Sloan Digital Sky Survey survey (25), and the Particle Physics Data Grid (21). The impact of these projects has been enormous in their respective fields. Over 3,000 astrophysics articles refer to the Sloan Digital Sky Survey alone. Yet, despite the significant advantages to be gained by sharing data, the high-performance computing community has lagged behind their colleagues in opening access to their datasets. We believe this is in part due to some lingering fundamental misconceptions about the nature of simulation and analysis. Once these issues are recognized, it becomes clear that opening access to high-performance computing datasets presents a major opportunity for the growth of computational sciences.

The first common misconception is that large-scale computational datasets which required the world's largest supercomputers to compute would also necessarily require the world's largest supercomputers to store and to analyze. However, this is not the case. For a wide class of problems which are CPU-bound (and not memory-bound), the results of large-scale simulations performed on the world's lagest machines can be analyzed on smaller machines, such as widely-available small-scale Linux clusters. As a concrete example, consider a traditional high-performance computing time-dependent simulation of a set of partial differential equations discretized explicitly onto a uniform mesh of N^D cells in D dimensions. In this case, due to the Courant limitation on the timestep, the number of CPU operations to complete the simulation will scale as N^{D+1}, even though the number of operations to be performed for a local analysis scales only as N^D. Based on this analysis, it is easy to see that local post-processing of a dataset will be roughly cheaper by a factor of N over the full simulation. In state-of-the-art 3D simulations, N approaches $(2-4)\cdot 10^3$, and the number of CPUs utilized during the full simulation range upwards of 10^5. However, simply scaling these numbers, it is clear that the post-processing analysis of this same dataset can be completed in a roughly equal amount of wall-clock time on just a few hundred processors. Significantly, *the analysis of wide classes of large-scale datasets can be completed on small-scale clusters.* In fact, all of the analysis shown in this paper was completed on several small-scale Linux clusters at the University of Chicago, all smaller than 256 processors.

Another common misconception is that distribution of large-scale datasets could possibly eat up large amounts of bandwidth, and take very long times to transfer. However, this is not a fundamental issue, either – for instance, one solution is simply to sidestep the transfer of the datasets and collocate storage and computation, allowing open access to the system to all interested parties. We believe quite strongly that this is in fact the natural solution for very large datasets, and have adopted the collocated model in establishing a new datastore system at the University of Chicago. Our turbulence dataset will be served to the community from the University of Chicago's Computation Institute's (CI) large datastore. The system currently is a scalable high-performance storage resource that has 75 TB of raw storage configured in an 8+2 RAID array, allowing up to 48 drives to fail with no impact to performance, stability, or reliability. It can deliver a sustained throughput of 3 GB/s. It can also be scaled to 480 TB of raw storage. Five I/O servers are connected to storage system by five fiber channels and then connected to the outside world via 1 Gb/s Ethernet connections to the CI's 10 Gb/s I-WIRE link. Collocated with the storage resource is the CI's TeraPort compute resource; a 244 processor AMD Opteron cluster that will be used for local processing of the data.

High-performance computing consists of a hierarchy of computation – at the top, the world's fastest machines, used by a handful of researchers, and towards the bottom, smaller machines accessible to a much wider community. The public release of the Flash Center dataset will amplify the impact of our turbulence effort significantly by sharing our results

with many levels on the computational hierarchy. Moreover, if our data dissemination effort is successful, we hope it will provide both a fruitful model and concrete software tools for other computational scientists to release their datasets in a similar fashion.

7. Conclusion

The ASC Flash Center at the University of Chicago has produced the largest weakly-compressible, homogeneous isotropic turbulence simulation to date. The simulation was performed on the world's largest and fastest supercomputer, BG/L, located at Lawrence Livermore National Laboratory. The results were produced using the newly released version 3 of the FLASH code, which is a modular, application-specific tool for astrophysical simulations which scales well to massively parallel environments. The BG/L configuration of over 64,000 processors with limited memory and computing power posed special challenges. Specialist input/output routines were developed, and efficient particle-tracking schemes were implemented. Despite the demanding conditions, preliminary analysis of the results indicates a dataset of very high quality. Extended self-similarity analysis on the data shows good support for new theories of turbulent structure and match well with experimental evidence. The massive dataset will be released to the public, thereby further expanding the impact of this simulation.

Acknowledgments

This work is supported in part at the University of Chicago by the U.S Department of Energy under Contract B523820 to the ASC/Alliances Center for Astrophysical Thermonuclear Flashes. The authors gratefully acknowledge the generous computational resources supplied by the DOE/ASC program at LLNL, as well as additional resources supplied by Argonne National Laboratory and IBM Watson.

In addition, the authors would like to thank a number of highly-talented individuals who supported our effort, and without whom this project would not have been possible. At LLNL, we thank Steve Louis, Adam Bertsch, Richard Hedges, Jean Shuler, Dick Watson, Mike McCoy, and the LC Computing Staff. At Argonne National Laboratory, we thank Ed Jedlicka and Susan Coghlan. At IBM we thank Bob Walkup, James Sexton, and Sameer Kumar.

Bios

Robert T. Fisher

Dr. Fisher received his Ph.D. in physics from the University of California at Berkeley in 2002, and did his first postdoc at the Lawrence Livermore National Laboratory. He is currently a member of the DOE/ASC FLASH Center astrophysics group, and a research scientist in the Department of Astronomy and Astrophysics at the University of Chicago. His research interests include the fundamental physics of turbulent flows, type Ia supernovae, star formation, numerical algorithms, and high-performance computing.

Leo P. Kadanoff

Leo Kadanoff received has Ph.D. in Physics from Harvard in 1960. He has been a Professor of Physics at the University of Illinois, Brown University, and the University of Chicago, where he presently has emeritus status. He is President of the American Physical Society. His research interests include the fundamental physics of turbulent flows, interface motion, applied mathematics, condensed matter physics, and the critical analysis of scientific computing.

Don Q. Lamb

Dr. Lamb is the Louis Block Professor in Astronomy & Astrophysics and the Enrico Fermi Institute at the University of Chicago. He is the director of the DOE ASC/Alliance Flash Center. He is a Fellow of the American Physical Society and the American Academy of Arts and Sciences. His current research interests include Type Ia supernovae, gamma-ray bursts, galaxy clusters, and high-performance computing.

Anshu Dubey

Dr. Dubey received her Ph.D. in computer science from Old Dominion University in 1993. She joined the University of Chicago as Research Associate in Astronomy and Astrophysics in 1993, and then moved to the Flash Center in 2002. At the Flash Center, she currently leads the code group that is responsible for design, development and support of the FLASH code.

Tomasz Plewa

Dr. Plewa is a senior research associate at the University of Chicago. He obtained his PhD in theoretical astrophysics from the Warsaw University and spent several years at the Max-Planck-Institut fuer Astrophysik, Garching, where he worked on development of adaptive methods in application to astrophysical flows. His research focuses on problems related to core-collapse and thermonuclear supernovae, mixing processes, laboratory astrophysics, code validation, and development of high-performance multiphysics simulation tools.

Alan Calder

Dr. Calder received his Ph.D. from Vanderbilt University in 1997. After that he held research positions at NCSA and the University of Chicago, and he is now a Senior Research

Scientist at SUNY, Stony Brook. His research interests are in supernovae and the physics therein, large-scale computing, and validation.

Fausto Cattaneo

Dr. Cattaneo received his Ph.D. in applied mathematics from Cambridge University in 1984. He subsequently worked as a postdoctoral associate at the University of Colorado (Boulder), and the University of Chicago. Currently he is jointly associate professor in the Department of Astronomy and Astrophysics and the Computation Institute at the University of Chicago, and is also affiliated with the DOE/ASC Alliance Flash Center. Dr. Cattaneo is also a computational scientist at Argonne National Laboratory. He is interested in computational fluid dynamics and magnetohydrodynamics.

Peter Constantin

Dr. Constantin received his PhD in 1981 at the Hebrew University of Jerusalem in Mathematics. His current interests are in active combustion, complex fluids, singularities in fluids and turbulence. He is currently Louis Block Professor at the University of Chicago.

Ian Foster

Dr. Foster received his PhD in computer science from Imperial College, London, in 1988. He is currently director of the Computation Institute at the University of Chicago and Argonne National Laboratory, and professor of computer science at the University. His research interests include distributed computing and computational science.

Michael E. Papka

Michael E. Papka is the deputy associate laboratory director for Computing and Life Sciences at Argonne National Laboratory and a Senior Fellow at the Computation Institute (a joint effort between the University of Chicago and Argonne National Laboratory) as well as the visualization group leader at the DOE/ASC Alliance Flash Center. His current interests are in the visualization and data analysis of large datasets.

Snezhana I. Abarzhi

Dr. Snezhana I. Abarzhi received her Ph.D. in theoretical physics and mathematics in 1994 from Landau Institute for Theoretical Physics, Moscow. Her current research interests include turbulence and turbulent mixing, multi-scale processes and nonlinear physics, and stochastic and applied analysis in fluid dynamics, plasmas and astrophysics.

Shimon M. Asida

Dr. Asida is a lecturer at the Racah Ins. of Physics of the Hebrew University of Jerusalem. He received his Ph.D. in physics from that university in 1998. He spent his postdoc at the University of Arizona at Tucson, and a sabbatical at the Flash Center at

the Department of Astronomy and Astrophysics at the University of Chicago. His research interests includes unstable flows, convection in stars, and type Ia supernovae modeling.

Paul M. Rich

Paul Rich received his MS in Computer Science from the University of Chicago in 2006, and is now a scientific programmer in the code group of the ASC FLASH Center. His research interests include high-performance computing, parallel architectures and terascale data analysis and post-processing.

Chad C. Glendening

Chad Glendening received a bachelor's degree from Yale and a master's degree from the University of Chicago. He is currently a member of the DOE ASC/Alliance Flash Center visualization group and a Ph.D. student in the University of Chicago Department of Geophysical Sciences.

Katie Antypas

Katie Antypas received her MS in computer science from the University of Chicago. She is currently a high-performance computing consultant at NERSC at the Lawrence Berkeley National Laboratory (LBNL). She previously worked in the code group at the DOE ASC/ Alliance Flash Center as a scientific programmer.

Daniel J. Sheeler

Dan Sheeler received his M.S. in Computer Science and his B.A. in Physics both from the University of Chicago. He currently works as a scientific programmer on the ASC FLASH project in the department of Astronomy and Astrophysics at the University of Chicago.

Lynn B. Reid

Dr. Reid is a scientific programmer at the FLASH Center in the Department of Astronomy and Astrophysics at the University of Chicago. She received her master's degree in applied mathematics from the University of Dundee in 1987 and a doctorate in environmental engineering from the Massachusetts Institute of Technology in 1996.

Brad Gallagher

Brad Gallagher is a Senior Programmer working in the DOE/ASC Flash visualization group in the department of Astronomy and Astorphysics at the University of Chicago. His main areas of interest are scientific visualization and the development of parallel tools to help analyze large datasets.

Shawn G. Needham

Shawn Needham received his M.S. in Computer Science from the University of Chicago and his B.A. in Psychology from Northwestern University. He currently works as the Systems Administrator for the ASC FLASH project in the department of Astronomy and Astrophysics at the University of Chicago.

REFERENCES

1.Argonne National Lab. Parallel-NetCDF: A high performance API for NetCDF file access. http://www-unix.mcs.anl.gov/parallel-netcdf/.

2.Argonne National Lab. ROMIO: A high-performance, portable MPI-IO implementation. http://www-unix.mcs.anl.gov/romio/.

3.ASC Flash Center, University of Chicago. FLASH3 users' guide. http://flash.uchicago.edu/website/codesupport/users_guide3/home.py, 2006.

4.R. Benzi, S. Ciliberto, R. Tripiccione, C. Baudet, F. Massaioli, and S. Succi. Extended self-similarity in turbulent flows. *Physical Review E (Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics)*, 48:R29–32, July 1993.

5.L. Biferale, G. Boffetta, A. Celani, B. J. Devenish, A. Lanotte, and F. Toschi. Multiparticle dispersion in fully developed turbulence. *Physics of Fluids*, 17:1701–+, Nov. 2005.

6.A. C. Calder, B. C. Curtis, L. J. Dursi, B. Fryxell, G. Henry, P. MacNeice, K. Olson, P. Ricker, R. Rosner, F. X. Timmes, H. Tufo, J. W. Truran, and M. Zingale, editors. *High-Performance Reactive Fluid Flow Simulations Using Adaptive Mesh Refinement on Thousands of Processors*, 2000. http://sc2000.org (Gordon Bell Prize).

7.A. C. Calder, B. Fryxell, T. Plewa, R. Rosner, L. J. Dursi, V. G. Weirs, T. Dupont, H. F. Robey, J. O. Kane, B. A. Remington, R. P. Drake, G. Dimonte, M. Zingale, F. X. Timmes, K. Olson, P. Ricker, P. MacNeice, and H. M. Tufo. On validating an astrophysical simulation code. *Astrophysical Journal, Supplement*, 143:201–229, Nov. 2002.

8.P. Colella and H. M. Glaz. Efficient solution algorithms for the Riemann problem for real gases. *Journal of Computational Physics*, 59:264–289, June 1985.

9.P. Colella and P. R. Woodward. The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of Computational Physics*, 54:174–201, September 1984.

10.Department of Energy. Secretary Abraham announces record breaking supercomputer performance. http://www.energy.gov/news/1537.htm, November 2004.

11.V. Eswaran and S. B. Pope. An examination of forcing in direct numerical simulations of turbulence. *Computers and Fluids*, 16:257–278, 1988.

12.U. Frisch. *Turbulence. The legacy of A.N. Kolmogorov*. Cambridge: Cambridge University Press, —c1995, 1995.

13.B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *Astrophysical Journal, Supplement*, 131:273–334, November 2000.

14.GDB. The GDB human genome database. http://www.gdb.org/.

15.T. Gotoh, D. Fukayama, and T. Nakano. Velocity field statistics in homogeneous steady turbulence obtained using a high-resolution direct numerical simulation. *Physics of Fluids*, 14:1065–1081, Mar. 2002.

16.Y. Kaneda, T. Ishihara, M. Yokokawa, K. Itakura, and A. Uno. Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box. *Physics of Fluids*, 15:L21–L24, Feb. 2003.

17.A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. *Royal Society of London Proceedings Series A*, 434:9–13, July 1991. English translation of Kolmogorov 1941.

18.A. G. Kritsuk, M. L. Norman, and P. Padoan. Adaptive mesh refinement for supersonic molecular cloud turbulence. *Astrophysical Journal, Letters*, 638:L25–L28, Feb. 2006.

19.P. MacNeice, K. M. Olson, C. Mobarry, R. de Fainchtein, and C. Packer. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126:330–354, April 2000.

20.NCSA. Heirarchical Data Format, version 5. http://hdf.ncsa.uiuc.edu/HDF5/.

21.PPDG. Particle Physics Data Grid. http://www.ppdg.net/.

22.E. Raymond. The cathedral and the bazaar. *Knowledge, Technology, and Policy*, 12(3):23–49, 1999.

23.Z.-S. She and E. Leveque. Universal scaling laws in fully developed turbulence. *Physical Review Letters*, 72:336–339, Jan. 1994.

24.The Globus Alliance. GridFTP. http://www.globus.org/grid_software/data/gridftp.php.

25.D. York, J. Adelman, J. Anderson Jr, S. Anderson, J. Annis, N. Bahcall, J. Bakken, R. Barkhouser, S. Bastian, E. Berman, et al. The Sloan Digital Sky Survey: Technical summary. *The Astronomical Journal*, 120(3):1579–1587, 2000.